IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

TITLE OF THE INVENTION:

Method and Apparatus for Creating and Executing Secure Scripts

INVENTOR:

RODNEY A. KORN

Prepared by

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN
12400 WILSHIRE BOULEVARD
SEVENTH FLOOR
LOS ANGELES, CA 90025-1026
(408) 720-8300

BACKGROUND OF THE INVENTION

Field of the Invention

The present invention relates to the field of encryption. Specifically, the present invention relates to creating and executing secure, i.e., encrypted, scripts by a world wide web-enabled application.

Description of the Related Art

Present World Wide Web browsers, such as Internet Explorer, available from Microsoft Corporation, are limited by the constraints of the HyperText Mark-Up Language (HTML). Web content based on HTML comprises static, two dimensional text and graphics. A scripting language, such as JavaScript - a cross-platform, object-based scripting language for client and server applications developed by Netscape Communications, Inc., extends a Web browser's capabilities. A scripting language allows access to objects within the browser and supports execution of Web applications. A script, written in a scripting language, typically has access to browser objects in an HTML document or page, and is capable of modifying variables in the HTML document. Thus, the script extends the capabilities of HTML processing without requiring interaction with a HyperText Transfer Protocol (HTTP) server. The script typically is downloaded by the browser as part of an HTML page and is processed as the page is received, or when a browser event occurs, such as the click of a button on the HTML page.

Sub A1

~~A script differs from an applet. Although an applet also is downloaded as part of a Web~~ page and run on a client system, the applet stands alone, that is, it is not part of the browser application, just as a an application program, such as a word processor application, is not part of ~~an operating system.~~

In addition to scripts and applets, controls enhance Web browsers. For example, ActiveX controls are interactive objects in a Web page that provide interactive and user-controllable functions. ActiveX controls are part of a set of technologies available from Microsoft Corporation, based on a refinement of the well known COM standard, that is directed to enabling interactive content for Web pages. ActiveX currently is supported by the Microsoft Windows operating system, but will be supported on other platforms, such as the Macintosh platform available from Apple Computer, and UNIX platforms.

Without sufficient security mechanisms in place, it is possible to download a Web page that contains controls that launch an application that causes harm or unintended results, e.g., to the client system. Furthermore, if the controls are not secure, the provider of a Web site risks attack by computer hackers, and is vulnerable to software bugs.

BRIEF SUMMARY OF THE INVENTION

The invention provides a method for creating a secure script. Executable commands in the script are hashed, and the hashed values for the commands are encrypted and appended to the script.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example and not limitation in the following figures. Like references indicate similar elements, in which:

Fig. 1 is a flow chart illustrating an embodiment of the invention.

Fig. 2 is a flow chart illustrating an embodiment of the invention.

Sub a2 >

DETAILED DESCRIPTION OF THE INVENTION

~~An embodiment of the present invention enables Web pages to execute software~~ applications on a client system, e.g., a personal computer (PC), in a secure manner using a signed control, and a signed and encrypted script. Embodiments of the invention may be represented as a software product received over, and/or stored on, a machine-readable medium (also referred to as a computer-readable medium or a processor-readable medium). The machine-readable medium may be any type of magnetic, optical, or electrical storage medium including a diskette, CD-ROM, memory device (volatile or non-volatile), or similar storage mechanism. Moreover, the machine readable medium may accessed at a server by a client via a network connection between the client and server, for example, in a client/server computing environment. The machine-readable medium may contain various sets of instructions, code sequences, configuration information, or other data. For example, the procedures described herein can be stored on the machine-readable medium. Those of ordinary skill in the art will appreciate that other instructions and operations necessary to implement the described invention may also be ~~stored on the machine-readable medium.~~

In one embodiment of the invention, a script in a World Wide Web page ("Web page", "Web document", or "HyperText Markup Language (HTML) document") is hashed and encrypted. A control in the Web page, such as ActiveX, decrypts and hashes the script to verify the script has not been altered or tampered with, before executing or causing to execute the script. In this manner, one can serve to a client web pages that contain interactive content or that execute local applications in a secure fashion. The described embodiment involves a script that may be invoked by a Web browser application, or more particularly, by a control in a Web page

downloaded by the Web browser application. However, it should be noted that any application or software program can benefit from the present invention to protect malicious modification of or hacking to a script or the like.

With reference to Fig. 1, the process starts at 110 with hashing the commands in the script. The script is written in a scripting language, such as JavaScript, and comprises executable commands to cause the client system upon which the script is executed to perform some function. The function may be defragmenting a hard disk drive accessible by the system upon which the script is executed, or providing interactive content in a Web page downloaded to a client system, e.g., online tutorial or help. The content of the script is not so important as is preventing unauthorized control of the script or unauthorized alteration of the script content in so much as an embodiment of the present invention is concerned.

Any well known or proprietary hashing function may be utilized to compute a hashed value for each executable command in the script. Each executable command is provided at 105 as the key value input to the hashing function, from which the hashing function computes a hashed value corresponding to the executable command. In one embodiment of the invention, each executable command may be hashed, while in other embodiments of the invention, some number of executable commands, e.g., one or more but less than all of the executable commands, may be hashed. In one embodiment of the invention, the hashing function utilizes public key A that is tied to the script, as described below, thus making it highly unlikely that the script was authored or edited by an unauthorized individual without access to the corresponding private key.

At 120, each hashed value is encrypted using well known asymmetric, i.e., public, key cryptography techniques. For example, each hashed value is encrypted using private key A 106.

This process is also referred to in cryptography as creating a public key digital signature. Public key digital signatures provide a way to prove that the signed data was signed by one who had a copy of a particular private key, in this case, private key A.

The signed hashed values for the executable commands are embedded or appended to the script at 130. Alternatively, the hashed values may first be appended to the script and then signed. A public key A corresponding to the private key A may be appended to the script as well, or obtained from the public key authentication infrastructure, e.g., a certification authority. (A public file known as a certificate is issued by the certification authority and contains an entity's public key, identifying information, and a signature provided by the certification authority). At 140, the script, including the signed hashed values and public key, if present, may be encrypted using a symmetric key 107 to provide a second level of encryption. The encryption is not necessary for protection of the script, but hides the public key, if included in the script.

In a Web-enabled application, the script, encrypted or not as the case may be, is converted as appropriate for inclusion in a Web page. The public key A 108 corresponding to the private key A 106 is provided to control, i.e., interactive objects that provide interactive and user-controllable functions, in the Web page. In one embodiment of the invention, the Web page utilizes ActiveX control from Microsoft Corporation. The control is also signed at 160, to hide public key A provided therein at 150. The control is signed using a different private key, key B provided at 109. The script is ready for the execution process upon activation of the control by, e.g., a Java applet or a user clicking a button on the Web page.

The process of securely executing the script is now described with reference to Fig. 2. In one embodiment of the invention, a user running a Web browser application visits a Web site and downloads a Web page containing interactive content. The user activates a control in the

Web page, for example, by clicking on an applet. Recall from the above discussion that the control is signed at 160 with a public key digital signature using private key B 109. Thus, at 210, the signature is verified using public key B 205. Verification is accomplished by decrypting the signed control with public key B. If any change has occurred to either the control or the signature, it will be detected at 210. At 220, the script is decrypted with symmetric key 107. (Symmetric key encryption requires only one key that is shared by the encryption process and decryption process). Of course, the decryption is necessary only if the script was correspondingly encrypted at 140.

At 230, the executable commands in the script are hashed, using the same hashing function utilized at 110. The hashed commands that were encrypted and appended to the script at 120 and 130, respectively, are now decrypted at 240, using public key A, which was provided to the control at 150. The decrypted hashed commands are compared at 250 with the commands hashed at 230. If no changes in the script occurred between hashing and encrypting at 110 and 120, and hashing and decrypting at 230 and 240, the decrypted hashed commands obtained at 240 should be identical to the hashed commands obtained at 230, and the script may begin execution at 260. If, on the other hand, the commands hashed at 230 are not the same as the hashed commands decrypted at 240, the user is cautioned or warned, for example, by displaying a message in a pop up window or the like in a display screen for the client system. The user may, according to one embodiment of the invention, select to proceed with execution of the script. This is useful, for example, if a new version of the script is released, in which case hashed values for the commands in the old version of the script will not match the hashed values for the commands in the new version.

In one embodiment of the invention, the decrypted hashed commands are maintained so that a comparison between hashed command values and decrypted hashed command values may be performed before every execution of the script. Alternatively, a comparison is performed between execution of each command, to ensure there is no dynamic modification of the script or particular commands in the script. In each case, the user is warned as appropriate. In this manner, verification of the source and integrity of a script in an application, such as may be in a Web page, is accomplished.